Research Article

Maxim A. Kuznetsov* and Ivan V. Oseledets

Tensor Train Spectral Method for Learning of Hidden Markov Models (HMM)

https://doi.org/10.1515/cmam-2018-0027
Received October 15, 2017; revised February 26, 2018; accepted May 2, 2018

Abstract: We propose a new algorithm for spectral learning of Hidden Markov Models (HMM). In contrast to the standard approach, we do not estimate the parameters of the HMM directly, but construct an estimate for the joint probability distribution. The idea is based on the representation of a joint probability distribution as an N-th-order tensor with low ranks represented in the *tensor train* (TT) format. Using TT-format, we get an approximation by minimizing the Frobenius distance between the empirical joint probability distribution and tensors with low TT-ranks with core tensors normalization constraints. We propose an algorithm for the solution of the optimization problem that is based on the alternating least squares (ALS) approach and develop its fast version for sparse tensors. The order of the tensor d is a parameter of our algorithm. We have compared the performance of our algorithm with the existing algorithm by Hsu, Kakade and Zhang proposed in 2009 and found that it is much more robust if the number of hidden states is overestimated.

Keywords: Multilinear Algebra, Tensor Train Decomposition, Alternating Least Squares (ALS), Hidden Markov Models (HMM), Spectral Algorithms

MSC 2010: 15A69, 65C40, 60J20

1 Introduction

Hidden Markov models (HMM) are important techniques in many applications. They are used in speech recognition [9, 21], natural language processing [16] and genomic sequence modeling [12] and many more. The HMM observable state x_n depends only on a current hidden state h_k, where $h_1, \dots, h_R$ form a classical Markov chain. The discrete HMM model is completely defined by the initial state distribution $\boldsymbol{\pi} \in \mathbb{R}^R$, a state transition probability matrix $\mathbf{T} \in \mathbb{R}^{R\times R}$ for the hidden states and an observation probability matrix $\mathbf{O} \in \mathbb{R}^{N\times R}$, where N is the number of observed states, and R is the number of hidden states. Our goal is to estimate the joint probability distribution

$$\mathbf{P}(i_1, \dots, i_N) = \mathbf{Pr}(x_1 = i_1, \dots, x_N = i_N), \tag{1.1}$$

given only sequential observations $x_1, \dots, x_N$ of a HMM. We will use the following operator representation of HMM, that is straightforward to verify (see for example [10])

$$\mathbf{P}(i_1, \dots, i_d) = \boldsymbol{\pi}^\top \mathbf{A}(i_1)^\top \dots \mathbf{A}(i_d)^\top \mathbf{1_r}, \tag{1.2}$$

***Corresponding author: Maxim A. Kuznetsov,** Skolkovo Institute of Science and Technology, Skolkovo Innovation Center Moscow, 143025, Moscow, Russia, e-mail: m.kuznetsov@skoltech.ru
Ivan V. Oseledets, Skolkovo Institute of Science and Technology, Skolkovo Innovation Center Moscow, 143025, Moscow, Russia, e-mail: i.oseledets@skoltech.ru

where

$$\mathbf{A}(i) = \mathbf{T}\,\mathrm{diag}(\mathbf{O}_{i,1}, \ldots, \mathbf{O}_{i,R}), \quad i = 1, \ldots, N.$$

is an $R \times R$ matrix for any fixed $i = 1, 2, \ldots, N$.

A classical approach is to estimate the set of parameters $\mathbf{T}, \mathbf{O}, \boldsymbol{\pi}$ from the data using, for example, the Expectation Maximization (EM) algorithm [3], which is a local search method that may fall into a local maximum.

Another approach is to estimate the joint probability distribution (1.1) directly, without estimating the parameters. Spectral algorithms have proven to be very efficient in this context, see [8, 18]. In this paper we propose a new method for learning joint probability distribution of HMMs from sequential observations that is based on applying *tensor-train decomposition* (TT-decomposition) as a low-parametric model. The main advantage of the proposed method is that it does not require any matrix inversions and does not have any restrictions on the number of observed and hidden states. We can even overestimate the number of hidden states R but our algorithm still produces a meaningful estimate.

We formulate the problem of joint probability estimation as a problem of tensor approximation via tensor networks, especially TT-format. Tensor methods have been already successfully applied to learning HMM (for example, see [1]), but in a different context, using *canonical polyadic (CP) tensor format* [4, 11]. The computation of the CP-decomposition is known to be generally unstable [5]. In this paper we use the TT-decomposition [13, 15] as our main tool, since it provides stable algorithms. It is also worth to note about the work [20], where a spectral algorithm based on the SVD (singular value decomposition) is proposed for latent tree graphical models, but it still involves inversion of matrices. The paper [19] is the closest to our concept. For a tree latent variable model it employs the *Hierarchical Tucker (HT)* decomposition [6, 7] to approximate the joint probability distribution. The TT-format is a special case of the HT-format, but in many practical applications the simplest choice of the tree is the most efficient one. A simple algebraic structure of the TT-format also significantly simplifies the development of new algorithms.

To summarize, our objective is to present an efficient algorithm for learning HMM using the TT-format which has polynomial complexity in n and r and is suitable for the estimation of the joint probability function. We also show its efficiency on synthetic examples with randomly generated HMM.

2 Hidden Markov Model (HMM) as a Tensor Train (TT)

A tensor of dimensionality d, $\mathbf{A}(i_1, \ldots, i_d)$, is said to be in the TT-format if its elements can be represented in the form

$$\mathbf{A}(i_1, \ldots, i_d) = \mathbf{G}_1(i_1) \ldots \mathbf{G}_d(i_d),$$

where $\mathbf{G_k}(i_k)$ is an $R_{k-1} \times R_k$ matrix, with $R_0 = R_d = 1$. For a given HMM, we associate with it a joint probability distribution $\mathbf{P}_d(i_1, \ldots, i_d)$ and treat it as a tensor. The number $d \le N$ is a parameter of our algorithm (recall that our final goal is to estimate the joint probability distribution defined by (1.1)), which defines the dimensionality of the joint probability distribution, and should be defined on the basis of reasons discussed below in Section 3.

Equation (1.2) justifies the TT-structure of the joint probability distribution of the HMM with additional structure: all the cores are equal to each other. In our approach, we lift this restriction and seek for a low-parametric representation of the joint probability distribution in the form

$$\mathbf{P}_d(i_1, \ldots, i_d) \approx \mathbf{G}_1(i_1) \ldots \mathbf{G}_d(i_d). \tag{2.1}$$

This increases the number of parameters to be estimated, but allows for nice and efficient algorithms. Equation (2.1) can be treated as a data model, that allows us to use presented algorithm in case when number of hidden states are unknown or probably overestimated as well and in principle the TT-based algorithm can be applied beyond the homogeneous HMM concept, because the TT-structure seems to be more general model than HMM.

3 Estimating TT-Model from the Data: Optimization Problem

Given sequential observations $x_1, \dots, x_N$ of the HMM, we can estimate the joint probability distribution as their relative frequencies from the observations:

$$\widehat{\mathbf{P}}_d(i_1, \dots, i_d) = \alpha \sum_{k=1}^{N-d+1} I[x_k = i_1, x_{k+1} = i_2, \dots, x_{k+d-1} = i_d], \tag{3.1}$$

where scaling coefficient α is chosen in such a way that

$$\sum_{i_1,\dots,i_d} \widehat{\mathbf{P}}_d(i_1, \dots, i_d) = 1$$

and $I[x_k = i_1, x_{k+1} = i_2, \dots, x_{k+d-1} = i_d]$ is indicator.

Using the TT-structure of $\mathbf{P}_d$, the estimate can be significantly improved. Among all possible tensors with such structure, we want to find the one that gives the best match in *Frobenius norm* to the empirical joint probability distribution tensor. Also, such tensor should correspond to a probability distribution, i.e. its elements should be non-negative and sum up to 1. This yields an optimization problem on the class of TT-tensors of rank R TT(R):

$$\min_{\mathbf{Z}} \|\widehat{\mathbf{P}_d} - \mathbf{Z}\|_F^2 \quad \text{subject to} \quad \mathbf{Z} \in \mathrm{TT}(R), \sum_{i_1,\dots,i_d} \mathbf{Z}(i_1, \dots, i_d) = 1. \tag{3.2}$$

Problem (3.2) needs a non-negativity constraint, but that it is typically numerically satisfied in the algorithm. The minimization problem (3.2) allows us to find an improved estimate $\mathbf{P_d}$ of the joint probability distribution tensor as follows. First, we estimate probabilities

$$\mathbf{P}_k(i_1, \dots, i_t) = \mathbf{P}_{k,t}(x_k = i_1, \dots, x_{k+t-1} = i_t)$$

by the following formula:

$$\hat{\mathbf{G}}_i = \sum_{j_i} \mathbf{G}_i(\alpha_{i-1}, j_i, \alpha_i) \quad \text{for all } i \in [d]$$

and

$$\mathbf{P}_1 = \mathbf{G}_1 \dots \mathbf{G}_t \hat{\mathbf{G}}_{t+1} \dots \hat{\mathbf{G}}_d.$$

Every $\mathbf{P}_i$, $i = 1, \dots, d - t + 1$, can be obtained in a similar way by cyclic shift of the matrices $\mathbf{G}_1, \dots, \mathbf{G}_d$. The final estimate reads

$$\widehat{\mathbf{P}}_d = \frac{\sum_l \mathbf{P}_l}{d - t + 1}.$$

There is a tradeoff between smaller and larger d. The larger d, the worse $\widehat{\mathbf{P}}_d$ approximates $\mathbf{P}_d$, but more estimates of $\widehat{\mathbf{P}}_d$ are available due to the gap $(d - t)$. The selection of optimal d is an interesting question that requires additional study.

4 Solution of the Optimization Problem

4.1 TT-SVD Algorithm

There are several ways to solve the optimization problem (3.2). If we for a second omit the normalization condition and consider just the problem of approximation of a given tensor by a tensor in TT-format, this can be done by the TT-SVD algorithm [13]. The computation of a quasi-optimal approximation is reduced to d singular value decompositions (SVD) of auxiliary matrices. The corresponding algorithm is summarized in Algorithm 1.

Algorithm 1. TT-SVD Algorithm.

```
1 Data: $d$-dimensional tensor $\mathbf{A}$, TT-ranks $R_k$.
  Result: TT-decomposition core tensors $G_1 \dots G_d$ corresponding to tensor $\mathbf{B} \approx \mathbf{A}$, with TT-ranks $R_k$.
1 {Initialization}
  Temporary tensor $C = A$
2 for $k = 1$ to $d - 1$ do
3     $C := \text{reshape}(C, [r_{k-1} n_k, \frac{\text{numel}(C)}{r_{k-1} n_k}])$.
4     Compute the singular value decomposition: $\mathbf{C} = \mathbf{USV}^\top$, and in the matrix $\mathbf{U}$ leave only $R_k$ singular vectors, corresponding to the $R_k$ largest singular values.
5     New core $\mathbf{G}_k := reshape(\mathbf{U}, [r_{k-1}, n_k, r_k])$.
6     $\mathbf{C} := \mathbf{SV}^\top$.
7 end
8 $\mathbf{G_d} = \mathbf{C}$.
```

To make the result of Algorithm 1 to be a probability distribution, we can just divide it by the sum of its elements. From numerical experiments we observed, that the result of the TT-SVD algorithm has negligibly small negative elements (if any), so explicit non-negativity constraint is not required. This fact requires an additional study and theoretical investigation. The main problem with the TT-SVD algorithm is its computational cost: it requires dense matrices with $\mathcal{O}(n^d)$ elements to be handled, and the final complexity can be estimated as $\mathcal{O}(n^{d+1} r^2)$, see [13] for details.

4.2 Removing Exponential Complexity: ALS Algorithm

An input tensor $\widehat{\mathbf{P}}$ is in fact a *sparse tensor* [2]: it has at most N non-zero elements (in practice, less), and this number is much smaller than n^d. Incorporation of sparsity in the TT-SVD algorithm is not a straightforward task, since the matrices will become denser and denser at each step and moreover, we need only left factors of the singular value decomposition which again is not easy to be done in the sparse format. Instead, we propose to use the *alternating least squares* (ALS) algorithm which is a standard tool in the computation of tensor decompositions.

The idea of ALS can be simply presented as follows:
(1) We fix all factors except k-th.
(2) Minimize (3.2) over $\mathbf{G}_k$, which reduces to a quadratic optimization problem.
(3) Cycle for all factors.

It is now well understood that the convergence of the ALS-type algorithms for the TT-format is much better than for the canonical format [14, 17]. Moreover, the ALS algorithm can be implemented in an cheap way. The TT-representation is non-unique, and the cores $\mathbf{G}_k(i_k)$ can be orthogonalized in two ways: *from the left* and *from the right*. The core $\mathbf{G}_k(i_k)$ can be stored as an $r_{k-1} \times n_k \times r_k$ tensor. Its left unfolding $\mathbf{G}_k^<$ is defined as an $r_{k-1} \times (n_k r_k)$ matrix obtained from this tensor by reshaping in Fortran order. Analogously, $\mathbf{G}_k^>$ is defined as an $(r_{k-1} n_k) \times r_k$ matrix obtained from $\mathbf{G}_k(i_k)$ by reshaping in Fortran order. To optimize over G_k, we first orthogonalize all cores $\mathbf{G}_1, \dots, \mathbf{G}_{k-1}$ from the left by sequential orthogonalization: First, compute the QR-factorization of the first factor $\mathbf{G}_1 = \mathbf{Q}_1^> \mathbf{R}_1$. The matrix $\mathbf{R}_1$ is then transferred to the second core, $\mathbf{G}_2'(i_2) = \mathbf{R}_1 \mathbf{G}_2(i_2)$, it is orthogonalized from the right: $\mathbf{G}_2'(i_2) = \mathbf{Q}_2^>(i_2)\mathbf{R}_2$, transfer $\mathbf{R}_2$ to the third core and so on. Analogously we could make the cores $\mathbf{G}_{k+1}, \dots, \mathbf{G}_d$ *right-orthogonalized*, yielding an equivalent representation of the form

$$\mathbf{Z}(i_1, \dots, i_d) = \mathbf{Q}_1^>(i_1) \dots \mathbf{Q}_{k-1}^>(i_{k-1}) \mathbf{G'}_k(i_k) \mathbf{Q}_{k+1}^<(i_{k+1}). \tag{4.1}$$

Equation (4.1) is a linear mapping from $\mathbf{G'}_k(i_k)$ to the space of D-dimensional tensors. It can be written in the matrix form

$$\|\widehat{\mathbf{p} - Q g_k}\| \to \min, \tag{4.2}$$

where $\widehat{\mathbf{p}}$ is a vector of length $(n_1 \cdots n_d)$, $\mathbf{g}_k$ is a vector of length $(r_{k-1} n_k r_k)$ and $\mathbf{Q}$ is a matrix of dimension $(n_1 \cdots n_d) \times (r_{k-1} n_k r_k)$. The prominent feature of the orthogonalization process is that the matrix $\mathbf{Q}$ will have orthonormal columns [13]. Then the solution of the optimization problem (3.2) without constraint reads

$$\mathbf{g}_k = \mathbf{Q}^\top \widehat{\mathbf{p}}.$$

Normalization constraint can easily be incorporated. In terms of the cores, the restriction

$$\sum_{i_1,\dots,i_d} \mathbf{Q}_1^>(i_1) \dots \mathbf{Q}_{k-1}^>(i_{k-1}) \mathbf{G}'{}_k(i_k) \mathbf{Q}_{k+1}^<(i_{k+1}) = 1$$

reduces to the equation

$$\sum_{i_k} \mathbf{\Phi}_k^> \mathbf{G}_k'(i_k) \mathbf{\Phi}_k^< = 1, \tag{4.3}$$

where $\mathbf{\Phi}_k^>$ is computed by summing each $\mathbf{Q}_s^>(i_s)$, $s = 1, \dots, k$, over i_s and taking the matrix-by-matrix product

$$\mathbf{\Phi}_k^> = \Big(\sum_{i_1} \mathbf{Q}_1^>(i_1) \Big) \dots \Big(\sum_{i_{k-1}} \mathbf{Q}_{k-1}^>(i_{k-1}) \Big).$$

The matrix $\mathbf{\Phi}_k^<$ is computed analogously. The constraint (4.3) can be rewritten as

$$(\mathbf{g}_k, \mathbf{v}) = 1, \tag{4.4}$$

where $\mathbf{v}$ is a vector of length $r_{k-1} n_k r_k$. The solution of the optimization problem (4.2) with constraint (4.4) reads

$$\mathbf{g}_k = \mathbf{Q}^\top \widehat{\mathbf{p}} - \alpha \mathbf{v},$$

where α is selected in such a way that $(\mathbf{g}_k, \mathbf{v}) = 1$.

The main computational complexity comes from the product $\mathbf{Q}^\top \widehat{\mathbf{p}}$. It can be rewritten in the index form (we go back from vectors to tensors)

$$\begin{aligned} \mathbf{G}_k(\alpha_{k-1}, i_k, \alpha_k) = \sum \widehat{\mathbf{P}}(i_1, \dots, i_d) \mathbf{Q}_1^>(i_1, \alpha_1) \dots \mathbf{Q}_{k-1}^>(\alpha_{k-2}, i_{k-2}, \alpha_{k-1}) \\ \times \mathbf{Q}_{k+1}^<(\alpha_{k+1}, i_{k+1}, \alpha_{k+2}) \dots \mathbf{Q}_d^<(\alpha_{d-1}, i_d), \end{aligned} \tag{4.5}$$

where the sum goes through $i_1, \dots, i_{k-1}, i_{k+1}, \dots, i_d$ and $\alpha_1, \dots, \alpha_{k-2}, \alpha_{k+1}, \dots, \alpha_d$.

If the elements of the tensor $\widehat{\mathbf{P}}$ are obtained as relative frequencies, it contains at most N non-zero elements. We store $\widehat{\mathbf{P}}$ as a sparse tensor in the coordinate format, i.e. we store the location of all non-zero elements and their frequencies. Suppose that there are $T \le N$ non-zero elements in $\widehat{\mathbf{P}}$. Then the summation in (4.5) can be replaced by the summation over all non-zero elements, and instead of n^d operations it will take T operations to compute a sum over $i_1, \dots, i_d$ and its corresponding contribution to $G_k(\alpha_{k-1}, i_k, \alpha_k)$. In our current implementation, we just compute an element by such summation, leading to an algorithm with complexity $C_{\text{ALS}} = N * C_{\text{step}} + C_{\text{tdot}} = \mathcal{O}(dr^3 n + Tdnr^2)$ operations.

5 Experiments

We tested and validated our method on a synthetically generated data. We initialize the matrices $\mathbf{O}$, $\mathbf{T}$, the vector $\boldsymbol{\pi}$ randomly, then we generated an HMM realization $x_1 \dots x_n$ of length $n = 10000$. We will compare our method with a spectral method described in [8]. Since we know the exact values of $\mathbf{O}$ and $\mathbf{T}$, the required probabilities can be computed exactly. As an accuracy measure, we use relative error ε in the Frobenius norm for tensors.

First we study the effect of order d on the accuracy of the estimates for different t. The results are presented in Figure 1.

In general, the number of hidden states is unknown, that is why r could be chosen incorrectly. The following experiment illustrates that the TT-method is stable with varying R and the spectral algorithm from [8] is unstable. Fix $N = 3$, $d = 7$, and "real" rank $\hat{R} = 2$, we will vary R as an algorithm parameter. The results are presented in Figure 2.

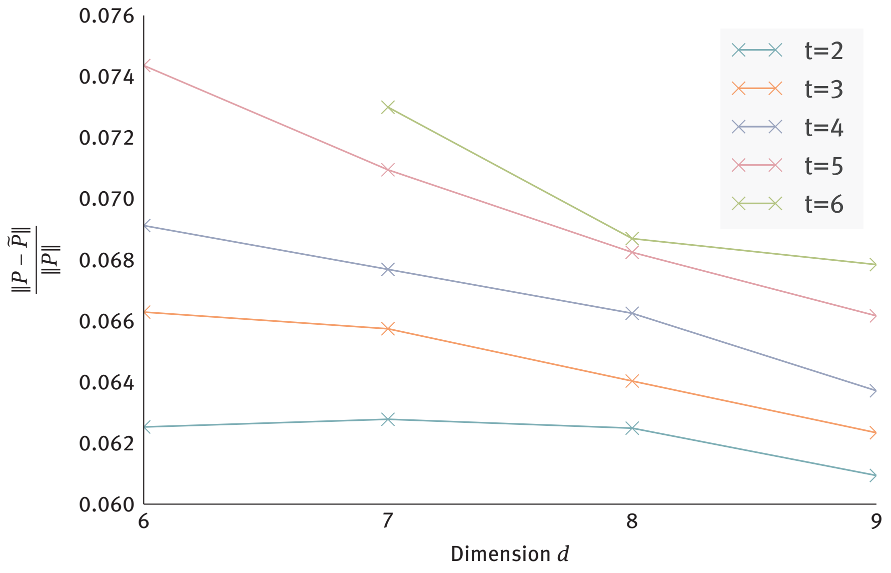

Figure 1: Accuracy of estimates of different marginal probability distributions $\mathbf{P}_{1,d}(x_1, \ldots, x_t)$ using the TT-algorithm with different values of d, $N = 3$, $R = 2$ for different t.

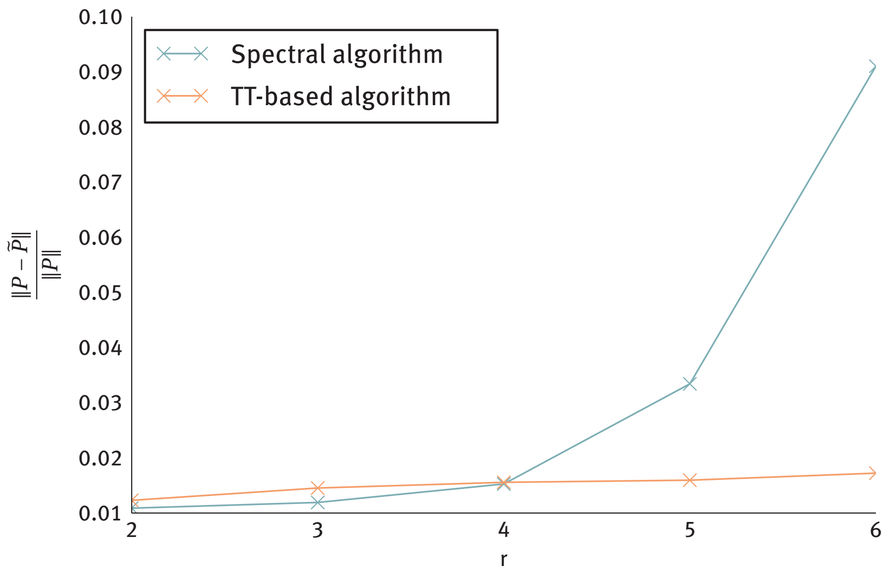

Figure 2: Dependence of the accuracy of the estimate from R with the true number of hidden states $\hat{R} = 2$, $N = 7$, $d = 5$, the spectral algorithm from the paper [8].

6 Conclusions

We consider the problem of estimation of a joint probability distribution for HMM using the tensor network approach. The solution has a low-rank TT-structure, and thus we formulate the optimization problem as a minimization problem over a manifold of tensors with bounded TT-ranks and propose an efficient algorithm to estimate the TT-cores, based on the ALS approach. Our algorithm uses only QR decompositions and matrix-by-matrix products and does not rely on inversion of matrices. We compared its efficiency and validated our algorithm on synthetic data with a spectral algorithm from [8] and found it comparable in terms of accuracy when the number of hidden states R is chosen correctly, but superior when it is overestimated. Moreover, the model for a joint probability distribution using the TT-format is more general than the Hidden Markov Model itself, since it allows to employ different numbers of hidden states at each step.

Acknowledgment: The authors express their deep gratitude to Professor Andrzej Cichocki for his helpful comments and assistance.

Funding: The authors gratefully acknowledge the financial support from Ministry of Education and Science of the Russian Federation under grant 14.756.31.0001.

References

[1] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade and M. Telgarsky, Tensor decompositions for learning latent variable models, *J. Mach. Learn. Res.* **15** (2014), 2773–2832.
[2] B. W. Bader and T. G. Kolda, Efficient MATLAB computations with sparse and factored tensors, *SIAM J. Sci. Comput.* **30** (2007/08), no. 1, 205–231.
[3] L. E. Baum, T. Petrie, G. Soules and N. Weiss, A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains, *Ann. Math. Statist.* **41** (1970), 164–171.
[4] J. D. Caroll and J. J. Chang, Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart–Young decomposition, *Psychometrika* **35** (1970), 283–319.
[5] V. de Silva and L.-H. Lim, Tensor rank and the ill-posedness of the best low-rank approximation problem, *SIAM J. Matrix Anal. Appl.* **30** (2008), no. 3, 1084–1127.
[6] L. Grasedyck, Hierarchical singular value decomposition of tensors, *SIAM J. Matrix Anal. Appl.* **31** (2009/10), no. 4, 2029–2054.
[7] W. Hackbusch and S. Kühn, A new scheme for the tensor representation, *J. Fourier Anal. Appl.* **15** (2009), no. 5, 706–722.
[8] D. Hsu, S. M. Kakade and T. Zhang, A spectral algorithm for learning hidden Markov models, *J. Comput. System Sci.* **78** (2012), no. 5, 1460–1480.
[9] X. D. Huang, Y. Ariki and M. A. Jack, *Hidden Markov Models for Speech Recognition*, Edinburgh University, Edinburgh, 1990.
[10] H. Jaeger, Observable operator models for discrete stochastic time series, *Neural Comput.* **12** (2000), no. 6, 1371–1398.
[11] T. G. Kolda and B. W. Bader, Tensor decompositions and applications, *SIAM Rev.* **51** (2009), no. 3, 455–500.
[12] A. Krogh, B. Larsson, G. Von Heijne and E. L. L. Sonnhammer, Predicting transmembrane protein topology with a hidden Markov model: Application to complete genomes, *J. Mol. Biol.* **305** (2001), no. 3, 567–580.
[13] I. V. Oseledets, Tensor-train decomposition, *SIAM J. Sci. Comput.* **33** (2011), no. 5, 2295–2317.
[14] I. Oseledets, M. Rakhuba and A. Uschmajew, Alternating least squares as moving subspace correction, preprint (2017), https://arxiv.org/abs/1709.07286.
[15] I. V. Oseledets and E. E. Tyrtyshnikov, Breaking the curse of dimensionality, or how to use SVD in many dimensions, *SIAM J. Sci. Comput.* **31** (2009), no. 5, 3744–3759.
[16] L. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, *Proc. IEEE* **77** (1989), no. 2, 257–286.
[17] T. Rohwedder and A. Uschmajew, On local convergence of alternating schemes for optimization of convex problems in the tensor train format, *SIAM J. Numer. Anal.* **51** (2013), no. 2, 1134–1162.
[18] S. M. Siddiqi, B. Boots and G. J. Gordon, Reduced-rank hidden Markov models, in: *International Conference on Artificial Intelligence and Statistics*, PMLR, (2010), 741–748.
[19] L. Song, M. Ishteva, A. Parikh, E. Xing and H. Park, Hierarchical tensor decomposition of latent tree graphical models, in: *Proceedings of the 30th International Conference on Machine Learning* (ICML-13), PMLR (2013), 334–342.
[20] L. Song, E. P. Xing and A. P. Parikh, A spectral algorithm for latent tree graphical models, in: *Proceedings of the 28th International Conference on Machine Learning* (ICML-11), PMLR (2011), 1065–1072.
[21] K. Stratos, M. Collins and D. Hsu, Unsupervised part-of-speech tagging with anchor hidden Markov models, *Trans. Assoc. Comput. Linguist.* **4** (2016), 245–257.